Handling Probabilistic Integrity Constraints in Pay-as-you-go Reconciliation of Data Models

Nguyen Quoc Viet Hung[a], Matthias Weidlich[b], Nguyen Thanh Tam[c], Zoltán Miklós[d], Karl Aberer[c], Avigdor Gal[e], Bela Stantic[a]

[a] *Griffith University, Gold Coast, Australia*
[b] *Humboldt-Universität zu Berlin, Germany*
[c] *École Polytechnique Fédérale de Lausanne, Switzerland*
[d] *Université de Rennes 1, France*
[e] *Israel Institute of Technology, Haifa, Israel*

Abstract

Data models capture the structure and characteristic properties of data entities, e.g., in terms of a database schema or an ontology. They are the backbone of diverse applications, reaching from information integration, through peer-to-peer systems and electronic commerce to social networking. Many of these applications involve models of diverse data sources. Effective utilisation and evolution of data models, therefore, calls for matching techniques that generate correspondences between their elements. Various such matching tools have been developed in the past. Yet, their results are often incomplete or erroneous, and thus need to be reconciled, i.e., validated by an expert. This paper analyses the reconciliation process in the presence of large collections of data models, where the network induced by generated correspondences shall meet consistency expectations in terms of integrity constraints. We specifically focus on how to handle data models that show some internal structure and potentially differ in terms of their assumed level of abstraction. We argue that such a setting calls for a probabilistic model of integrity constraints, for which satisfaction is preferred, but not required. In this work, we present a model for probabilistic constraints that enables reasoning on the correctness of individual correspondences within a network of data models, in order to guide an expert in the validation process. To support pay-as-you-go reconciliation, we also show how to construct a set of high-quality correspondences, even if an expert validates only a subset of all generated correspondences. We demonstrate the efficiency of our techniques for real-world datasets comprising database schemas and ontologies from various application domains.

Keywords: data integration, probabilistic constraints, model reconciliation

1. Introduction

Data models are an important means to design, analyse, and improve systems that process structured data. Assuming a broad view on how the structure and characteristic properties of data entities can be captured, we assume a generic view on such models. That includes, for instances, database schemas, which describe the structure of data entities in terms of their attributes along with containment hierarchies and functional dependencies between them. Another example are ontologies that ground the interpretation of data entities in well-defined semantics, e.g., through concepts and generalisation dependencies. The commonality of schemas and ontologies is that both provide a vocabulary of terms in a particular domain, and enforce consistent use of these terms via structure relations, such as tree-like representations [1, 2]. They differ, though, in how they enforce this consistency. For example, schemas define data types, whereas ontologies use logical systems [1, 2].

Regardless of the assumed notion of a data model, their creation, utilisation, and evolution is supported by manifold techniques that offer, for instance, re-use driven modelling support, harmonisation of model variants, model-based system validation, and effective management of model repositories. Many of these techniques built upon the identification of correspondences between the elements of data models. For schemas, this process is known as *schema matching*. Similarly, *ontology alignment* is the process of linking ontological concepts by correspondences. While the specific techniques differ depending on the type of data model (schema or ontology), they typically rely on the notion of correspondences to connect schema attributes or ontology entities [1, 2]. As such, the accuracy and, thus, usefulness of techniques supporting the creation, utilisation, and evolution of data models depends on the correctness and completeness of model matching.

There is a large body of work on model matching techniques. Numerous commercial and academic matching tools, called *matchers*, have been developed in recent years to generate correspondences between pairs of database schemas [1, 3, 4] and ontologies [5, 6]. Even though matchers achieve impressive performance for some datasets, they cannot be expected to yield a correct result in the general case. Since matchers rely on heuris-

Email addresses: quocviethung.nguyen@griffith.edu.au (Nguyen Quoc Viet Hung), matthias.weidlich@hu-berlin.de (Matthias Weidlich), tam.nguyenthanh@epfl.ch (Nguyen Thanh Tam), zoltan.miklos@univ-rennes1.fr (Zoltán Miklós), karl.aberer@epfl.ch (Karl Aberer), avigal@technion.ac.il (Avigdor Gal), b.stantic@griffith.edu.au (Bela Stantic)

tic techniques, their result is inherently uncertain. In practice, therefore, model management in general and model matching in particular include a post-matching phase, in which correspondences are reviewed and validated by an expert [7, 8, 9]. This step is referred to as *reconciliation* of the data models [10], based on the result of automated model matching.

Matching Networks. In this work, we focus on a setting in which matching is conducted for a set of related models. The vast majority of existing matchers generate correspondences between *pairs* of data models. Hence, matching a set of related models requires the repeated application of a matcher to generate a *matching network*, spanned by the correspondences between elements of the respective models. Such a network, however, shall meet consistency expectations in terms of global integrity constraints. Correspondences generated by a matcher for a pair of data models in isolation may turn out to be problematic when considering the network-level. Correspondences between one pair of models may be inconsistent with correspondences between other models.

The presence of network-level integrity constraints imposes challenges for manual validation of correspondences in a matching network. Constraints introduce dependencies between the validation of different correspondences, which are hard to understand especially in large-scale networks. At the same time, experts have a limited effort-budget, so that complete reconciliation of a network of correspondences is typically not a feasible option. As such, there is a need to generate a trusted set of correspondences that is largely in line with the integrity constraints, based on expert input on the correctness of solely a subset of the correspondences in the network.

Reconciliation of Matching Networks. To address the above challenges, an expert may be guided in their reviewing efforts, identifying the correspondences for which validation is most beneficial. Specifically, for each correspondence that has not yet been validated, network-level integrity constraints enable the computation of a probability of the correspondence being correct [11]. Intuitively, a correspondence is assigned a high correctness probability, if it does not lead to violations of integrity constraints, regardless of the correctness of other non-validated correspondences. By defining an information-theoretic model of network uncertainty over these probabilities, guidance of an expert with a limited effort budget can then be phrased as an optimisation problem [11].

Existing approaches to reconciliation of matching networks, however, put forward a simplistic view on data models. Such models are defined as sets of model elements [11], which neglects their internal structure, e.g., containment hierarchies and functional dependencies between attributes of a database schema or generalisation relations defined for the concepts of ontologies. At the same time, existing formulations of integrity constraints are based on Boolean formalisms [12, 13], such as Answer Set Programming. This implicitly assumes that the models in a matching network capture data entities at the same level of abstraction. Only in that case, integrity constraints can be expected to be satisfied by *all* correspondences of the network. Once some models differ in their applied abstraction of some data entities, integrity constraints may be satisfied by the vast majority of correspondences, but a also violated by a few of them.

Handling Probabilistic Integrity Constraints. In this paper, we argue that a more realistic view on data models in matching networks is needed, one that incorporates the internal structure of data models and copes with inconsistencies stemming from different abstraction levels of data models. To cater for such a setting, we present a model of probabilistic integrity constraints, for which satisfaction is preferred, but not required. In addition to the definition of this model, our contributions are reasoning methods that enable pay-as-you-go reconciliation of matching networks under probabilistic integrity constraints:

- We show how to compute the correctness probabilities of correspondences in a matching network under probabilistic integrity constraints using the graphical formalism of a factor graph. To cope with computational challenges, we also introduce methods to deal with large-scale data and incremental updates.
- We develop a method to instantiate a trusted set of correspondences under probabilistic integrity constraints. We show that this instantiation can be formulated as an optimisation problem and propose a heuristic to approximate a solution to this problem.

The remaining part of the paper is structured as follows. The next section provides the background for our work. That includes a discussion of different types of data models, their associated matching and reconciliation problems, and a general framework to guide experts in the reconciliation of matching networks. In Section 3, we present a novel formal model for matching networks with probabilistic integrity constraints. Using this model, Section 4 shows how to compute the correctness probabilities of correspondences, while our approach to instantiation of a trusted set of correspondences is introduced in Section 5. Section 6 gives details on our experimental evaluation. Finally, Section 7 summarizes related work, before Section 8 concludes the paper.

2. Background

To provide background for our work, we first elaborate on matching of data models in Section 2.1. We then turn to the intuition of matching networks in Section 2.2. Finally, we summarise the state-of-the-art in terms of a framework for pay-as-you-go reconciliation of matching networks in Section 2.3.

2.1. Matching of Data Models

In this work, we explore a generalisation of different matching problems that have been described in the literature for specific types of data models. Specifically, we review the problems of schema matching and ontology alignment.

Schema Matching. Schema matching is the process of generating correspondences between the attributes of two database schemas, for the purpose of some data integration task [14, 15]. An example is the often quoted coffee consumption data found in Google Fusion Tables [16, 17], which is distributed among different tables that represent a specific region [18]. Extraction

of information over all regions requires means to query and aggregate across multiple tables, thereby raising the need of an integrated view of the data. Further applications that require schema matching include:

- *Corporate data:* Databases of large enterprises are often developed independently to cater for particular requirements imposed by legal regulations, business models, or value chains. Hence, data resides in multiple sources in an enterprise. To support cross-database queries, the schemas of the databases need to be matched [19, 20].
- *P2P networks:* A P2P network is a decentralized and distributed architecture in which participating nodes act as peers that share data. Searching in these networks requires means to query across multiple peers, and schema matching helps to overcome the heterogeneity typically observed in P2P networks [21].
- *Cloud platforms:* Cloud applications enable storage and processing of data distributed across PCs, mobile devices, and online services. [1] [2] [3] To realize a unified view on such data, see [22], schema matching supports the horizontal integration of data across different cloud solutions.

Solutions to the matching problem for database schemas have been developed for decades, see [1] for a survey.

Ontology Alignment. Ontologies enable the definition of semantics of data entities. An ontology can be viewed as a vocabulary to define models of a particular domain, making explicit the relations between the underlying concepts. Ontologies developed for independent data sources, however, show differences in the syntactic, terminological, and conceptual representation of concepts. This motivates *ontology alignment*, which aims at generating semantic correspondences between the concepts of ontologies [5] for various use cases:

- *Product catalogues:* In business-to-business applications, online portals and shopping sites store information about their products in electronic catalogues [23, 24, 25]. However, the ontology used to describe products is often designed differently among sellers. To create a common market place, ontology alignment identifies correspondences between the concepts used to describe products.
- *Web services:* Ontologies provide a rich and precise language to describe the functionality of Web services that expose data via programmable interfaces for information search and discovery [26]. Yet, data and services of different providers are described in terms of diverse ontologies. In this context, ontology alignment helps to identify correspondences between service interfaces for the purpose of web service discovery and comparison [27].

Many matchers developed for schema matching have been adapted to support ontology alignment, e.g., COMA++ [28], YAM [29], and Harmony [30]. They are complemented by dedicated matchers for ontologies—see the studies conducted by the Ontology Alignment Evaluation Initiative (OAEI) [6].

[1] `https://one.ubuntu.com`
[2] `http://www.eyeos.com`
[3] `https://www.dropbox.com`

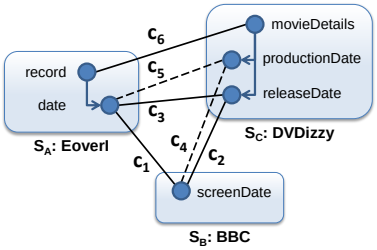

Figure 1: A matching network of real-world data models.

2.2. Matching Networks

The above applications, regardless of the type of data model, have in common the need to match sets of related models. Matchers developed for database schemas or ontologies proceed pair-wise, constructing correspondences between elements (attributes or concepts, respectively) of two models. Applying these matchers repeatedly for pairs of models induces a matching network that is spanned by correspondences between elements of the respective models. Such an approach enables on-the-fly integration of data models: The integration of a new model into an existing network is simply achieved by pair-wise matching the new model with existing ones.

An Example Matching Network. To give the intuition of matching networks, we consider the scenario of three online video content providers, EoverI, BBC, and DVDizzy. Each provider runs a Web portal, which enables potential customers to search for content (e.g., based on title or release date). Now, we consider the case that the three providers would like to offer their content via a shared marketplace. Then, as a first step, the databases storing information about the content need to be integrated. The structure of each of these databases is described by a schema and matching these schemas pair-wise creates correspondences between their attributes. Taking the real-world forms of the above mentioned video content providers as a representation of the underlying database schemas, a simplified view on a matching network is shown in Figure 1. It lists some attributes (record, date, screenDate, movieDetails, productionDate, and releaseDate) of the three schemas, some structural dependencies between them (date is part of record, while productionDate and releaseDate are part of movieDetails), and correspondences identified by pair-wise automated matching.

Network-level Integrity Constraints. The result of automatic matching of data models is inherently uncertain and it is widely acknowledged that general-purpose matching algorithms cannot be expected to yield results that are always correct. In particular, since matchers identify correspondences between pairs of models, the identification of correspondences is agnostic to global consistency expectations regarding the matching network. These expectations can be formalised as integrity constraints that, unlike those extracted from experts' knowledge [31], are domain-independent. Examples of network-level integrity constraints include the 1-1 constraint and the cycle constraint [12, 13].

2.3. Pay-As-You-Go Reconciliation

Due to the inherent uncertainty of automatic matching, correspondences are reviewed and validated by an expert user, a pro-

cess known as reconciliation [10]. Yet, most existing approaches study the reconciliation of the results of automatic matching at a pair-wise level. That is, correspondences between the elements of a pair of models are considered in isolation [7, 8, 9]. Such a pair-wise approach is ineffective for matching networks, since network-level integrity constraints introduce dependencies between the validation of different correspondences. As such, validation of some correspondences is particularly beneficial, as it may allow for immediate conclusions on other correspondences based on the integrity constraints. Exploiting such dependencies becomes even more important when only a subset of all correspondences in a network are validated, which is the common case of having only a limited effort-budget for the validation.

Against this background, it was suggested to guide experts in their validation efforts by computing expected benefit of validating a certain correspondence. Specifically, in [11], a pay-as-you-go reconciliation framework was introduced, which is summarised in Figure 2. The input of this framework is given by a set of candidate correspondences, which are generated by automatic matchers for a set of data models. Later integration of data models is facilitated, though: For a model that is added later, conducting pair-wise matching will simply yield further candidate correspondences.

For a given set of candidate correspondences, the general reconciliation process works as follows.

First, a matching network is constructed from the initial set of candidate correspondences. As part of that, a correctness probability is assigned to each correspondence. It is computed by exploiting the dependencies induced by the integrity constraints of the matching network and by incorporating the validation results from an expert (if available).

Based on the correctness probabilities of correspondences, an information-theoretic model enables guidance of an expert. That is, the uncertainty of a matching network is quantified based on the entropy over these correctness probabilities. Then, correspondences can be ranked based on the information gain of their validation, i.e., the reduction of uncertainty in the matching network if an expert approves or disapproves the correspondence. The selection of a set of beneficial correspondences then becomes an optimisation problem. Once expert assertions (approvals or disapprovals) are available, they are incorporated in the matching network: The correctness probabilities of the validated correspondences are set to one or zero, respectively, and the effect on all other correctness probabilities is computed.

At any point in time, however, a reconciliation result may be instantiated. Based on the current matching network, a maximal subset of correspondences that are likely to be both correct and in line with the integrity constraints is selected. This subset is referred to as the trusted matching that reflects the result at the current state of reconciliation.

The above framework defines enables effective guidance of experts in the reconciliation of matching networks. The network is updated incrementally by means of *probability computation*, whereas *matching instantiation* enables the derivation of a trusted set of correspondences at any point in time.

Existing methods to realise probability computation and matching instantiation, however, are limited in both, the formalisms adopted for data models as well as for integrity constraints. Specifically, in [11], data models are assumed to be merely sets of unrelated elements. Such a model ignores that data models often possess some internal structure, may it be containment hierarchies and functional dependencies as in the case of database schemas or generalisation relations between ontology concepts. Such structure shall be considered in the assessment of integrity, as discussed above for Figure 1.

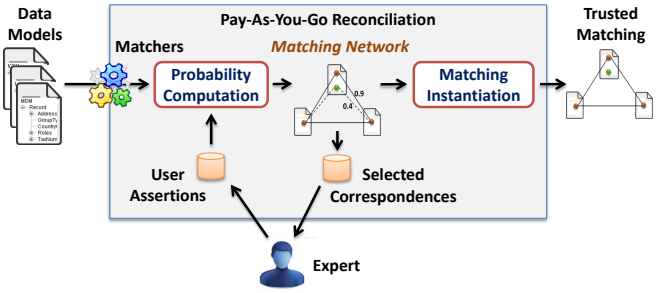

Figure 2: A generic view on pay-as-you-go reconciliation.

In the same vein, existing formalisms for integrity constraints, see [12, 13], are Boolean, i.e., a constraint is either satisfied or violated. Such a constraint model is inappropriate in many application scenarios. In particular, if models in a matching network capture data entities on different levels of abstraction, one cannot expect the integrity constraints to be satisfied by all correspondences.

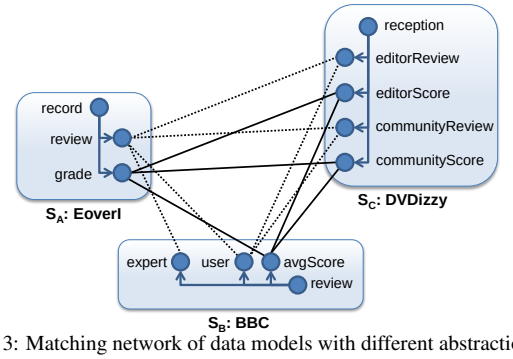

Figure 3: Matching network of data models with different abstraction levels.

Consider the matching network in Figure 3, which shows a different excerpt of the database schemas introduced already in Figure 1. The three schemas capture movie reviews and evaluation scores at different levels of abstraction. In schema S_A, the respective information is modelled by means of two attributes, review and grade. In schema S_C, reviews and scores are captured by different attributes, depending on whether they originate from editors (e.g., editorScore) or the community (e.g., communityScore). The correspondences shown in Figure 3 highlight that such differences in the applied abstraction level may lead to violations of integrity constraints. Both, the 1-1 constraint and the cycle constraint, as introduced above, are violated. Yet, arguably, this violation is due to the differences in how the schemas capture the domain.

To cope with such phenomena, we argue that a suitable formalism for integrity constraints shall enable some flexibility in the assessment. That is, consistency expectations shall be formulated as preferences rather than Boolean criteria. This

way, a model would capture that, in general, satisfying the 1-1 constraint and the cycle constraint can be expected to yield high quality correspondences, while at the same time acknowledging that a few correspondences may violate them.

3. Matching Networks with Probabilistic Constraints

Having introduced the intuition of matching networks in the previous section, this section presents a formal model for matching networks with probabilistic integrity constraints.

Data Models. We capture a *data model* as a tuple $\langle A, R\rangle$ with $A = \{a_1, ..., a_n\}$ being a set of *elements* and $R \subseteq A \times A$ as a *structure* relation defined over these elements. As such, we capture the essence of a model in terms of its defining elements and its structure, yet largely abstracting from the peculiarities of diverse types of data models and their representation languages. For instance, elements and the structure relation can be interpreted in terms of database schemas, i.e., elements are attributes, while the structure relation denotes a containment hierarchy or functional dependencies. In the domain of ontologies, in turn, elements denote concepts and the structure relation can capture a part-of relation between them. In general, data models may show a more fine-granular structure that is captured by further relations, e.g., ontologies may define not only a part-of relation, but also specialisation relations. For the sake of a compact formalisation, we limit the discussion to a single relation between the elements of a model. Our approach of considering this relation in the definition of integrity constraints, however, may be lifted directly to additional relations.

A matching network is defined for a finite set of data models $S = \{\langle A_1, R_1\rangle, \ldots, \langle A_m, R_m\rangle\}$, that are built of unique elements, i.e., $A \cap A' = \emptyset$ for $\langle A, R\rangle, \langle A', R'\rangle \in S$ and $A \neq A'$. Considering elements of various data models, we introduce a short-hand notation: We write $a \diamond a'$ if $a, a' \in A$, i.e., if two elements belong to the same data model. Furthermore, $A_S = \bigcup_{\langle A,R\rangle \in S} A$ and $R_S = \bigcup_{\langle A,R\rangle \in S} R$ denote the union of elements and relations in S respectively.

In the example in Figure 1, it holds $S = \{S_A, S_B, S_C\}$. Model S_C comprises elements $A_{S_C} = \{\mathsf{movieDetails}, \mathsf{productionDate}, \mathsf{releaseDate}\}$ and the structure relation $R_{S_C} = \{(\mathsf{movieDetails}, \mathsf{productionDate}), (\mathsf{movieDetails}, \mathsf{releaseDate})\}$.

Interaction Graphs. In some application scenarios, not all given data models are matched pair-wise when constructing a matching network, e.g., due to business requirements or privacy policies. Formally, we capture this aspect by means of an *interaction graph*, an undirected graph $G_S = (V, E)$, such that edges indicate which pairs of models in S need to be matched.

Revisiting the example in Figure 1, the interaction graph is defined as $G_S = (\{S_A, S_B, S_C\}, \{\{S_A, S_B\}, \{S_A, S_C\}, \{S_B, S_C\}\})$.

Correspondences. A *correspondence* represents a (semantic and/or syntactic) relation between model elements [1]. Formally, we write

$$\mathscr{A} = \bigcup_{\langle A,R\rangle, \langle A',R'\rangle \in S} \; \bigcup_{a \in A,\ a' \in A'} \{a, a'\}$$

to denote the set of all possible correspondences, i.e., all two-sets of elements of distinct models. Then, $\{a, a'\} \in \mathscr{A}$ denotes an individual correspondence. Note that even though many matchers generate solely simple one-to-one correspondences, our formulation does not preclude handling of one-to-many or many-to-many relations, which may be represented by the Cartesian product of the respective elements.

We refer to correspondences generated by automatic matchers as *candidate correspondences* since there is no guarantee that they are indeed correct [32, 33]. Given two distinct models $\langle A, R\rangle, \langle A', R'\rangle \in S$, we write $C_{A,A'} \subseteq \mathscr{A}$ to denote the set of all candidate correspondences returned by automatic matchers.

In the example in Figure 1, $c_1 = \{\mathsf{date}, \mathsf{screenDate}\}$ is one of the illustrated correspondences.

Expert Input. We model *expert input* as a tuple $U = \langle U^+, U^-\rangle$ of assertions, where $U^+ \subseteq \mathscr{A}$ and $U^- \subseteq \mathscr{A}$ are sets of approved and disapproved correspondences, respectively. That is, after expert input has been sought for a candidate correspondence $c \in C$, the assertions U are updated, yielding either $\langle U^+ \cup \{c\}, U^-\rangle$ (c is approved) or $\langle U^+, U^- \cup \{c\}\rangle$ (c is disapproved). In this work, we assume a feedback model, where expert input is considered to be correct, such that correspondences in U^+ must be included in a any trusted matching derived from the matching network, whereas correspondences in U^- must be excluded.

Probabilistic Integrity Constraints. A finite set $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ models the integrity constraints that formalize consistency expectations regarding the matching network, such as the 1-1 constraint or cycle constraint. We formalize network-level integrity constraints based on a probabilistic model. Given a set of candidate correspondences C, this model defines the probability of a constraint being satisfied. However, to capture the dependencies between constraint satisfaction and correspondences on a fine-granular level, this probability is not defined *globally* for the matching network, but *locally* for a particular set of correspondences. That is, an integrity constraint is encoded as a probability $P(\pi \mid C)$, which represents the probability that a set of possible correspondences $\pi \subseteq \mathscr{A}$ satisfies the constraint given a set of candidate correspondences C. We exemplify this model using several example constraints. Two of them are probabilistic generalisations of the Boolean constraints introduced in [12, 13]. A third one refers to the structure of data models, an aspect that has been ignored in existing reconciliation approaches.

- *1-1 constraint.* Let $\pi = \{\pi_1, \ldots, \pi_k\} \subseteq \mathscr{A}$ be a set of possible correspondences. Given a set of candidate correspondences C, the 1-1 constraint for π, denoted by $\gamma_{1\text{-}1}(\pi)$, is satisfied with probability

$$P(\gamma_{1\text{-}1}(\pi) \mid C) = \begin{cases} 1 & \text{if } \forall\, \{a, a'\} \in \pi : \\ & |\{\{a, a''\} \in C \cap \pi \mid a' \diamond a''\}| \leq 1 \\ \Delta & \text{otherwise} \end{cases} \tag{1}$$

  where $\Delta \in [0, 1]$ is a relaxation parameter and $\Delta = 0$ yields a hard constraint. Intuitively, the 1-1 constraint enforces that each element of one model should be matched to at most one element of any other model. In Figure 1, for instance, the set of candidate correspondences is given

as $C = \{c_1, c_2, c_3, c_6\}$. Here, the set of correspondences $\pi = \{c_3, c_5\} =$ {{date, productionDate}, {date, releaseDate}} violates the 1-1 constraint. Attribute date of schema S_A is matched to both productionDate and releaseDate of schema S_C, which correspond to the elements a' and a'' in the above definition.

- *Cycle constraint.* Let $\pi = \{\pi_1, \ldots, \pi_k\} \subseteq \mathscr{A}$ be a set of possible correspondences that form a cycle of data models, i.e., $\pi_j = \{a_j, a_{j+1}\}$ for $1 \leq j < k$ and $\pi_k = \{a_k, a_1\}$. Given a set of candidate correspondences C, the cycle constraint for π, denoted by $\gamma_{\circlearrowleft}(\pi)$, is satisfied with probability

$$P(\gamma_{\circlearrowleft}(\pi) \mid C) = \begin{cases} 1 & \text{if } \pi \subseteq C \\ 0 & \text{if } |\pi \setminus C| = 1 \\ \Delta & \text{otherwise} \end{cases} \quad (2)$$

  where $\Delta \in [0,1]$ is a relaxation parameter modelling the probability of compensating errors (i.e., two or more incorrect correspondences yielding a correct cycle). The cycle constraint enforces consistency in the sense that any cycle formed by correspondences is closed [12]. Revisit our example in Figure 1, with $C = \{c_1, c_2, c_3, c_6\}$. Here, the set of correspondences $\pi = \{c_1, c_2, c_5\} = \{$\{date, productionDate\}, \{date, screenDate\}, \{screenDate, releaseDate\}, \{productionDate, releaseDate\}$\}$ yields a violation: There is a path of correspondences between different elements (attributes productionDate and releaseDate) of a single model (schema S_C). This path denotes a violation as correspondence c_5 does not exist in C, i.e. $|\pi \setminus C| = 1$.
- *Structure constraint.* Let $S = \{\langle A_1, R_1\rangle, \ldots, \langle A_m, R_m\rangle\}$ be a set of data models. Let $\pi = \{\pi_1, \ldots, \pi_k, \pi'_1, \ldots, \pi'_k\} \subseteq \mathscr{A}$, $k < m$, be a set of possible correspondences, such that:
  - the correspondences form two paths, $\pi_j = \{a_j, a_{j+1}\}$ and $\pi'_j = \{a'_j, a'_{j+1}\}$ for $1 \leq j < k$;
  - the paths visit the same models, $a_j, a'_j \in A_j$ for $1 \leq j \leq k$;
  - the structure relation in the first and last model is consistent, $(a_1, a'_1) \in R_1$ and $(a_{k+1}, a'_{k+1}) \in R_{k+1}$.

  Given a set of candidate correspondences C, the structure constraint for π, denoted by $\gamma_R(\pi)$, is satisfied with probability

$$P(\gamma_R(\pi) \mid C) = \begin{cases} 1 & \text{if } \pi \subseteq C \\ 0 & \text{if } |\pi \setminus C| = 1 \\ \Delta & \text{otherwise} \end{cases} \quad (3)$$

  where $\Delta \in [0,1]$, again, is a relaxation parameter modelling the probability of compensating errors. The intuition behind the structure constraint is that the internal structure of data models shall be preserved by the correspondences. In Figure 1 example, the set of correspondences $\pi = \{c_3, c_6\}$ is arguably in line with the containment relations defined by model S_A (date is part of record) and model S_C (releaseDate is part of movieDetails).

Relaxation parameters Δ of constraints may be provided by application experts or be set based on an adaptive learning strategy. While this parameter configuration is not the focus of this paper, we outline a simple strategy to learn the parameter Δ for each constraint γ, as follows. The main idea is that the more violations are introduced by an expert, the more the associated constraints should be hardened; and vice-versa. Initially, we set $\Delta = 0.5$, since the integrity constraints do not affect the correctness of validated correspondences. Then, periodically, e.g., after obtaining further 20 expert assertions, we compute the set of constraint violations on top of all correspondences approved by the expert. Denote by $V = \{v_1, \ldots, v_n\}$ the union set of all constraint violations. For each violation $v_i \in V$, we count the number of correspondences in this violation. Then for each constraint γ involved in V, we set its new parameter Δ to the average value of the percentages of its violations.

Probabilistic Matching Networks. Combining the above notions, we define a *probabilistic matching network* as a tuple $N = \langle S, G_S, \Gamma, C, P\rangle$, where S is a set of data models, G_S is an interaction graph defined between these models, Γ is a set of integrity constraints, C is a set of candidate correspondences, and P is a probability model. The latter assigns a probability $P(c)$ to each candidate correspondence $c \in C$, indicating how likely it is that c is correct. This model integrates the user assertions U: since expert input is assumed to be correct, the probabilities of asserted correspondences are one or zero.

Trusted Matching. As detailed in Section 2.3, reconciliation aims at improving the quality of the matching network by means of expert input. However, in pay-as-you-go reconciliation, one may derive a *trusted matching* at any point in time. It denotes a maximal subset of correspondences that are likely to be correct and in line with the integrity constraints. In our model, such a trusted matching is denoted by $M \subseteq C$, i.e., a subset of the candidate correspondences. It can be seen as an approximation of the unknown ground truth based on the current knowledge about the correctness of correspondences.

4. Uncertainty Computation under Probabilistic Constraints

Following the general approach to pay-as-you-go reconciliation outlined in Section 2.3, a first step is the construction of a probabilistic matching network. Here, the starting point is given by a set of candidate correspondences that are derived by pair-wise matching of a given set of data models. For each such correspondence, we then need to determine the probability of it being correct. Since the confidence values commonly returned by automatic matchers are not normalized and often unreliable [1], we ground the computation of correctness probabilities for correspondences in the integrity constraints defined for the matching network. To this end, we adopt a model in which a correspondence is a random variable. Then, integrity constraints express dependencies between these random variables and assertions obtained by expert input are evidence for their truth values. Below, we first show how this idea is formalized using the model of a factor graph, before we turn to the actual computation of probabilities for the correspondences and scalability considerations.

4.1. The Factor Graph of a Matching Network

We capture the dependencies between the random variables that represent correspondences, integrity constraints, and expert input by means of a probabilistic graphical model, namely a *factor graph* [34]. In general, a factor graph establishes a relation between functions (called factors) defined over potentially overlapping sets of random variables. The model enables self-configuration when new information becomes available, which is an important asset to support pay-as-you-go reconciliation: With the arrival of new expert input, data models, or candidate correspondences, the model is updated incrementally by adding variables and factors.

A factor graph is a bipartite graph $\langle V, F, E \rangle$, where V is a set of random variables or evidence, F is a set of functions (factors), and $E \subseteq \{\{v, f\} \mid v \in V, f \in F\}$ are undirected edges. A set of random variables V and a set of factors F fully characterizes a factor graph. The definition of the edges relates each factor $f(v_1, \ldots, v_d) \in F$ to the random variables over which it is defined, i.e., $\{f, v_i\} \in E$ for $v_i \in V$, $1 \leq i \leq d$.

In our context, there are three types of random variables representing candidate correspondences, constraints, and expert input. We overload notation and use C, Γ, and U to refer to the actual correspondences, constraints, and expert assertions, as well as the associated random variables, i.e., $V = C \cup \Gamma \cup U$ defines the variable nodes of the factor graph. Further, the model includes correspondence factors f_C, constraint factors f_Γ, and expert factors f_U to encode relations between the variables, i.e., $F = f_C \cup f_\Gamma \cup f_U$ defines the factor nodes of the factor graph.

Correspondence Variables. As mentioned above, each correspondence $c \in C$ is assigned a random variable, also denoted by $c \in \{0, 1\}$, that indicates the correctness of the correspondence (the value 1 denotes correctness).

Constraint Variables (Evidence). We refer to a set of correspondences that violate or satisfy an integrity constraint as constraint evidence. Each of these sets is assigned a variable node $\Pi \subset C$ in the factor graph.

Expert Variables. Expert input $u \in U$ is directly considered as an (observed variable) $u \in \{0, 1\}$ (the value 1 denotes approval).

Correspondence Factors. Each correspondence variable c is associated with a prior-distribution factor $f_c : \{c\} \to [0, 1]$ that is determined either in a training phase or stems from automatic matchers (e.g., based on the confidence value assigned by these matchers). If no information is available, we start with $f_c(c) = 0.5$ following the maximum entropy principle. The set of correspondence factors is $f_C = \bigcup_{c \in C} f_c$.

Constraint Factors. A constraint factor node *cf* connects a correspondence to the constraint violations it involves. As a result, the correspondences are dependent on each other through multiple factors, which encodes their stochastic dependency. For illustration, we rely on the three aforementioned constraints. Let $\pi = \{\pi_1, \ldots, \pi_k\} \subseteq \mathscr{A}$ bet a set of possible correspondences. Then, constraint factors are defined as follows:

- The function for a factor that represents the *1-1 constraint* π is defined as:

$$cf_{\gamma_{1\text{-}1}(\pi)}(c_1, \ldots, c_k) = P(\gamma_{1\text{-}1}(\pi) \mid C) \tag{4}$$

  where $c_i \in \pi \cap C$ and $c_i \neq c_j$ for $1 \leq i, j \leq k$ and $i \neq j$.
- If the correspondences in π form a cycle (see, Equation 2), the factor representing the *cycle constraint* is defined as:

$$cf_{\gamma_{\circlearrowright}(\pi)}(c_1, \ldots, c_k) = P(\gamma_{\circlearrowright}(\pi) \mid C) \tag{5}$$

  where $c_i \in \pi \cap C$ and $c_i \neq c_j$ for $1 \leq i, j \leq k$ and $i \neq j$.
- If the correspondences in π form two inconsistent paths (see, Equation 3), the factor representing the *structure constraint* is defined as:

$$cf_{\gamma_R(\pi)}(c_1, \ldots, c_k) = P(\gamma_R(\pi) \mid C) \tag{6}$$

  where $c_i \in \pi \cap C$ and $c_i \neq c_j$ for $1 \leq i, j \leq k$ and $i \neq j$.

Note that the above probability computation is also applicable for the special case of a Boolean formulation of integrity constraints, as presented in [12, 13]. That is, the probability of such constraints becomes zero or one.

Expert Feedback Factors. To incorporate expert input, each correspondence variable c is connected with an expert input u via a factor node f_u. This factor directly encodes the response of the expert to accept or reject the respective correspondence:

$$f_u(c) = \begin{cases} 1 & \text{If } u = 1 \wedge c = 1 \\ 1 & \text{If } u = 0 \wedge c = 0 \\ 0 & \text{Otherwise} \end{cases} \tag{7}$$

Under a different model for expert input, this formulation can be adapted to include experts that are not fully reliable. However, such extended formalizations are beyond the scope of this work. We refer readers to [35, 36] for possible implementations of such adapted models.

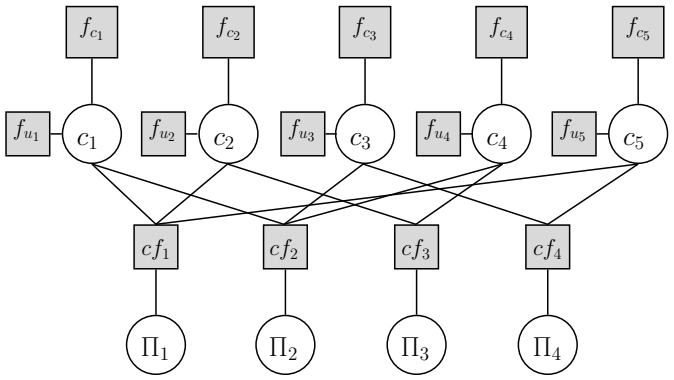

Figure 4: Factor graph representation of a matching network for the example given in Figure 1.

Taking up the example matching network of Figure 1, Figure 4 illustrates a respective factor graph. It comprises variables (shown as circles) for five correspondences $c_1, \ldots, c_5$ and four variables for evidence $\Pi_1, \ldots, \Pi_4$, each representing a set of

possible correspondences that violate one of the integrity constraints. For instance, $\Pi_1 = \{c_1, c_2, c_5\}$. Expert variables are not visualized. The figure also illustrates various factors, i.e., $f_{c_1}()$ is the correspondence factor assigning a prior for the correctness of correspondence c_1 to the respective random variable. Factor $f_{u_1}()$ links the latter to the observed variable of expert input for this correspondence (the variable is not visualized). Further, $cf_1()$ is a constraint factor that connects the set of possible correspondences Π_1 to the variables of the correspondences c_1, c_2, and c_5, implying that $\{c_1, c_2, c_5\}$ is a violation of a constraint—specifically, the cycle constraint is violated in this example. Note that the correspondence c_6 is not visualized here since it is not involved in any constraint violation; i.e., it is represented by an isolated variable with only a correspondence factor $f_{c_6}()$ and a feedback factor $f_{u_6}()$ connected to it.

4.2. Computing the Probabilities for Correspondences

The model of a factor graph enables us to compute the certainty of a correspondence in a matching network. This computation exploits the (marginal) probabilities of the random variables representing the correctness of correspondences. Following the model introduced above, correspondence variables are binary, so that $P(c = 1)$ (or $P(c)$ for short) is the probability that a correspondence $c \in C$ is correct. The computation of this probability is grounded in the correlations defined by the factor functions that relate the random variables to each other.

Various techniques are available to compute probabilities in a factor graph, most commonly *belief propagation* or *sampling*. The former considers the (un)certainty as information that is propagated through the factor graph and relies, for instance, on message-passing algorithms [34]. Yet, it has been observed that belief propagation converges slowly, if the graph is large and contains cycles. When reconciling matching networks, the number of variables grows quickly and cyclic dependencies become the rule, rather than the exception. To cope with large and dense factor graphs, we therefore resort to sampling to find the most probable values of random variables. Specifically, Gibbs sampling proved to be a highly efficient and effective mechanism for factor graphs [37].

We adopt the Expectation-Maximization (EM) principle [38] for probability computation on top of Gibbs sampling. This design choice is motivated by the fast convergence and computational efficiency of EM. We name our version as *s-EM*. In general, we compute the probability of correspondences and produce Gibbs samples mutually. Given a model parameter W, the probability computation is iterative and alternates between an *Expectation step* (E-step) and a *Maximization step* (M-step), until convergence. Therefore, we obtain a sequence $\Omega^0, \Omega^1, \ldots, \Omega^t$ of Gibbs samples and a sequence $P^0, P^1, \ldots, P^t$ of probabilities assigned to correspondences.

- *E-step:* derives a sequence of samples Ω^t by performing Gibbs sampling according to the probability distribution:

$$q(C) \propto \prod_{f \in F} P^{t-1}(c) f(.) \tag{8}$$

  where $f(.)$ are factor functions as described in Section 4.1. Note that Ω^t is a sequence, as any configuration of C can appear multiple times.
- *M-step:* Based on the current Gibbs samples, for each correspondence c, we estimate its probability:

$$P^t(c) = \frac{\sum_{C \in \Omega^t} \mathbb{1}_{c=1}}{|\Omega^t|} \tag{9}$$

If the difference between two consecutive estimates of all probabilities is insignificant, the process converges.

Continuing the example factor graph of Figure 4, we assume that the last Gibbs sampling comprised three configurations/samples (the number of samples is adjustable and the samples must not violate any integrity constraints), $\omega_1 = [1,0,0,1,1]$, $\omega_2 = [0,1,1,0,0]$, $\omega_3 = [1,0,0,1,0]$, where the i-th vector element denotes the correctness of correspondence c_i. Under this sampling result $\Omega = \{\omega_1, \omega_2, \omega_3\}$, we estimate the correctness probabilities of correspondences by Equation 9, for example, $P(c_1) = 2/3$ and $P(c_2) = 1/3$.

Guarantee 1. *The update time of* s-EM *is linear.*

Proof. The E-step is implemented by Gibbs sampling, which takes linear time [39, 40] w.r.t. the number of correspondences. The M-step also takes linear time to compute probability of each correspondence. □

In brief, given a probabilistic matching network formalised as $N = \langle S, G_S, \Gamma, C, P \rangle$, probability computation yields, for each $c \in C$, a probability value $P(c)$ indicating the likelihood of correspondence c to be correct. Note, again, that the factor graph model encodes all the information given by automatic matchers (in terms of priors) and expert input (in terms of assertions).

4.3. Scalability Considerations

In general, computing probabilities in the whole factor graph is an expensive computational task, which stands in contrast to the low response times needed in reconciliation based on expert input. However, reconciliation is an incremental process, meaning that only a few changes have to be incorporated at a time. Hence, recomputing the whole graph is typically not necessary once new expert input has been received. Following this line, an implementation of probability computation shall incorporate the following two techniques to achieve scalability:

- *Incremental Gibbs sampling:* The computation of Gibbs sampling can be adapted to proceed incrementally [41], rejuvenating the existing probability values in light of new data. Then, the new data is propagated to neighbouring nodes, while applying a decay function. The latter limits the consequences of a local change, and thus makes sampling faster.

  To illustrate this idea, we take up the example of Section 4.2. If a user approves $c_1 = 1$, the Gibbs sample ω_2 becomes invalid, so that we derive $P(c_1) = 1$. We conclude that maintaining the samples is beneficial for incremental computation as it avoids to perform the sampling from scratch.

- *Network modularity:* Even experts are typically overwhelmed when presented with a complete matching network. However, regardless of the type of data model, we observed that, in practice, these models are typically built from disjoint groups of model elements. Thus, candidate correspondences often relate to disjoint subsets of model elements, so that graph decomposition techniques [42] can be applied to decompose a large network into smaller ones to handle them more efficiently.
  In cases the disjoint groups are still large, the number of correspondences and constraint violations can grow fast, making it intractable to construct the factor graph. To handle such cases, decomposition strategies for partitioning a matching network into smaller parts can be applied [43]. The resulting partial networks can be efficiently transformed into factor graphs, with the trade-off of information loss regarding constraint violations that involve correspondences between elements of data models in different partitions. Yet, such information loss may be minimized by tracing the decomposition back to hypergraph partitioning. We note, though, that even for the large datasets used in our experimental evaluation (see Section 6), the construction of a factor graph did not turn out to be problematic and no decomposition of the matching network had to be applied.
  Consider the example of Figure 4 and assume that we shall decompose the network into two components. Then, the partitioning $\{c_1, c_2, c_3, c_4\}$, $\{c_5\}$ yields the smallest information loss. Only the violation Π_4 is not considered, since c_3 and c_5 do not connect to Π_4 anymore as they belong to different components.

5. Instantiation under Probabilistic Constraints

A distinguishing feature of pay-as-you-go reconciliation is the fact that a matching can be instantiated at all times, even if the matching network is not fully reconciled. Instantiating such a matching is particularly important for applications that value a fast setup time above waiting for full validation [44]. Also, various applications explicitly require a deterministic matching, e.g., to query a collection of data models, see Section 2.

In this section, we first formulate an optimisation problem for the instantiation of a trustful matching, i.e., a matching that comprises candidate correspondences that are most likely correct and in line with the specified integrity constraints. Since this problem turns out to be computationally costly, we then propose a heuristic-based algorithm to construct a near optimal solution.

5.1. The Instantiation Problem

Given a matching network, a set of candidate correspondences is called a *matching instance*, or *matching* for short. Ideally, a matching is the ground truth, which would be obtained after all candidate correspondences have been validated by an expert. Yet, this is impractical in most cases, so that a probabilistic matching network induces a set of matching instances that approximate the ground truth. To assess the quality of a particular matching instance $I \subseteq C$ of a probabilistic matching network $N = \langle S, G_S, \Gamma, C, P \rangle$, we consider three dimensions:

- *Violation Degree:* A matching instance of high quality should be likely to satisfy the integrity constraints. Formally, we capture this requirement by the *violation degree* per integrity constraint $\gamma \in \Gamma$, a function $v_\gamma : 2^C \to [0,1]$ that denotes the probability of a set of correspondences to violate the constraint. We exemplify the definition of this function for the 1-1 constraint formalized in Section 3. Let $I \subseteq C$ be a matching instance. Then, the violation degree is the probability of violating the constraint of the matching instance, i.e., $v_{\gamma_{1\text{-}1}}(I) = (1 - P(\gamma_{1\text{-}1}(I) \mid I))$.
- *Size:* A matching instance should relate a large number of model elements to each other. Given a matching instance $I \subseteq C$, its size in terms of the number of contained correspondences $|I|$ is a straightforward measure for this quality dimension.
- *Likelihood:* A matching instance should comprise correspondences that are likely to be correct. Therefore, we consider the *likelihood* of matching instances, which is defined by a function $u : 2^C \to [0,1]$. Given a matching instance $I = \{c_1, \ldots, c_k\} \subseteq C$, it captures the joint probability of the correspondences, $u(I) = P(c_1, \ldots, c_k)$. It is worth noting that using the factor graph representation of a probabilistic matching network, this joint probability is computed via the associated factors.

Using these measures, we model instantiation of a matching as an optimisation problem. When matching data models as detailed in Section 2, we prioritise the violation degree, since any instantiated matching shall be consistent. However, given the probabilistic nature of constraints, we adopt some tolerance threshold $\theta \in [0,1]$ for the violation degree. From all matching instances that show a violation degree that is less than the threshold, we identify one that has maximal size and maximal likelihood. Formally, the problem is described as follows.

Problem 1 (Matching Instantiation)**.** *Let $N = \langle S, G_S, \Gamma, C, P \rangle$ be a probabilistic matching network and $\theta \in [0,1]$ be a tolerance threshold. The problem of* matching instantiation *is the identification of a matching instance $I \subseteq C$ that satisfies the following conditions, in the descending order of priority:*

i) θ-satisfaction for all constraints: for all constraints $\gamma \in \Gamma$, it holds that $v_\gamma(I) \leq \theta$.
ii) Maximal size: for all matching instances $I' \subseteq C$, $I' \neq I$, that show θ-satisfaction for all constraints holds that $|I| \geq |I'|$.
iii) Maximal likelihood: for all matching instances $I' \subseteq C$, $I' \neq I$, that show θ-satisfaction for all constraints and maximal size holds that $Q(I) \geq Q(I')$.

An exact solution to the matching instantiation problem is called a *trustful matching*. Note that the problem is grounded solely in the correctness probabilities of correspondences, since expert input is incorporated in the probabilistic model.

Solving the problem of matching instantiation requires knowledge about the integrity constraints in the network. Unfortunately, even under the simplistic 1-1 constraint and even without the maximal likelihood condition, the instantiation problem is computationally hard.

Theorem 1. *Let $N = \langle S, G_S, \Gamma, C, P \rangle$ be a probabilistic matching network, such that $\Gamma = \{\gamma_{1\text{-}1}\}$ defines only the 1-1 constraint. Then, given a tolerance threshold $\theta \in [0,1]$ and an integer $\delta \in \mathbb{N}$, the problem of deciding whether there exists a matching instance of N that shows θ-satisfaction of Γ and is of size larger than δ is NP-complete.*

Proof. To prove the NP-completeness of our decision problem, we show that: (i) it is in NP and (ii) it is NP-hard.

Given a matching instance I, one can check in polynomial time whether its size is larger than δ and whether I shows θ-satisfaction of Γ (using the factor graph representation). Hence, (i) holds true.

Next, we argue that there is a polynomial time reduction of the *maximum independent set* (MIS), which is NP-complete, to our problem. MIS requires the identification of a maximal set of vertices in a graph $G = (V, E)$ such that no two vertices are adjacent. We construct a probabilistic matching network as follows: each vertex $v \in V$ is a correspondence. An edge $\{v_i, v_j\} \in E$ is represented by a pair of distinct correspondences that do not show θ-satisfaction of the 1-1 constraint, i.e., $\{v_i, v_j\} \in E$ iff $v_i, v_j \in \{c \in C \mid (1 - P(\gamma_{1\text{-}1}(c) \mid I)) > \theta\}$. This construction requires iterating over all pairs of nodes, i.e., polynomial time. Then, solving our decision problem yields a solution to MIS. Hence, (ii) holds true. □

5.2. A Heuristic Solution to Matching Instantiation

In light of Theorem 1, we develop a heuristic solution to find an approximation of a trustful matching efficiently. The approximate solution is found in polynomial time, yet may be non-optimal regarding size and likelihood.

Developing a heuristic solution for the problem of matching instantiation is challenging due to the complex dependencies between correspondences that are induced by the integrity constraints. Some correspondences always go together, whereas others are mutually exclusive because of the integrity constraints. These dependencies create a non-uniform joint distribution incorporating all possible matching instances. Our approach, therefore, is to rely on a randomized local search. The main idea is to keep exploring the neighbours of recent matching instances until termination (in our case, an upper bound of iterations), and record the one with the best size and likelihood.

Overview. Our approach to heuristic matching instantiation is formalised in Algorithm 1. It takes a probabilistic matching network, an upper bound for the number of iterations, and a tolerance threshold as input. It returns the best matching instance found during the search. Technically, the algorithm starts with a trivial matching instance that contains all correspondences for which the assigned probability is equal to one. This instance is then extended until the termination condition is satisfied (line 5). In each iteration, we first consider a set of remaining correspondences and their probabilities. One of these correspondences is added to the current matching instance I based on Roulette wheel selection. Once a correspondence has been added, the current matching instance may no longer show θ-satisfaction for all constraints. Therefore, the *adjust* function (defined below) potentially removes problematic correspondences from I to ensure θ-satisfaction (line 9). However, a correspondence could be added to I and then removed immediately by the *adjust* function. In that case, I would be left unchanged and the algorithm would be trapped in a local optima. Therefore, we employ the Tabu search method that uses a 'tabu' (forbidden) set of correspondences, so that the algorithm does not consider these correspondences repeatedly (line 6-8). Finally, a matching instance H is returned by evaluating the size and likelihood of matching instances explored so far.

Algorithm 1: Heuristic matching instantiation

input : a probabilistic matching network $N = \langle S, G_S, \Gamma, C, P \rangle$,
an upper bound for the number of iterations k,
a tolerance threshold θ
output : a matching instance H

```
   // Initialization
1  H ← {c ∈ C | P(c) = 1} ;
2  I ← H;
3  i ← 0 ;
4  T ← ∅ ;
5  while i < k do
       // Fitness proportionate selection
6      ĉ ← RouletteWheel_c({⟨c, P(c)⟩ | c ∈ (C \ I \ T)});
7      I ← I ∪ {ĉ};
8      T ← T ∪ {ĉ};
       // Adjust matching, so that it shows θ-satisfaction
9      I ← adjust(P, I, ĉ, Γ, θ) ;
       // Keep track of the best instance
10     if |H| < |I| then H ← I ;
11     if |H| = |I| ∧ Q(H) < Q(I) then H ← I ;
12     i ← i + 1;
13 return H
```

Adjusting a Matching Instance. Algorithm 2 details function *adjust* in Algorithm 1, which adjusts a matching instance that does not show θ-satisfaction. The key idea is to greedily remove correspondences that are involved in likely constraint violations, until the matching instance shows θ-satisfaction (line 2). We do so by first extracting all subsets of correspondences of the matching instance that contain the correspondence that has just been added ($\hat{c}$) and are problematic in terms of the constraints (line 3). We then identify correspondences that may be removed, because their correctness probability is less than one and that have not been just added (line 4). For these correspondences, we count how often they are part of subsets of correspondences that do not show θ-satisfaction (line 5) and remove the correspondences for which the largest count is obtained (line 7). The greediness of this approach is motivated by the idea that correspondences that are likely to cause the absence of θ-satisfaction of the original matching instance are removed first. This way, many of the correspondences of the matching instance are retained.

Properties of the Heuristic Solution. For the presented heuristic solution, we provide guarantees in terms of correctness and runtime performance.

Guarantee 2. *Algorithm 1 terminates and is correct.*

Proof. Termination follows from the upper bound k for the iteration in Algorithm 1 and the fact that in each iteration in Algorithm 2, one correspondence is removed, but none is added. Correctness follows from the following points: (i) A new correspondence is chosen from probable correspondences (line 6). (ii)

Algorithm 2: Adjusting a matching instance

input : a probabilistic model P,
a matching instance I,
an added correspondence $\hat{c}$,
a set of integrity constraints Γ,
a tolerance threshold θ
output : a matching instance $\hat{I}$ that shows θ-satisfaction for all constraints Γ

1 $\hat{I} \leftarrow I$
2 **while** $\exists\, \gamma \in \Gamma : v_\gamma(\hat{I}) > \theta$ **do**
  `// Get all sets of problematic correspondences containing` $\hat{c}$
3 $W \leftarrow \{C' \subseteq \hat{I} \mid \hat{c} \in C' \wedge \exists\, \gamma \in \Gamma : v_\gamma(C') > \theta\}$;
4 $I_P \leftarrow \{c \in \cup_{C' \in W} \mid c \neq \hat{c} \wedge P(c) < 1\}$;
  `// For each correspondence, count in how many problematic sets it occurs`
5 **for** $c \in I_P$ **do** $b_c \leftarrow |\{C' \in W \mid c \in C'\}|$;
  `// Greedily remove the one that occurs in the largest number of problematic sets`
6 $c^* \leftarrow \text{argmax}_{c \in I_P} b_c$;
7 $\hat{I} \leftarrow \hat{I} \setminus \{c^*\}$;
8 **return** $\hat{I}$

When adding a correspondence $\hat{c}$ to I (line 7) leads to absence of θ-satisfaction of the matching instance I, I is adjusted immediately (line 9). (iii) H always maintains the instance that is best in terms of size and likelihood. Thus, the algorithm yields a near-optimal solution to the problem of matching instantiation. □

Finally, we observe that the presented heuristic solution indeed allows for efficient instantiation of a matching for the aforementioned integrity constraints (see Section 3). For the 1-1 constraint and cycle constraint, the algorithm requires quadratic time in the number of candidate correspondences, which, as we demonstrate in our experimental evaluation, is tractable.

Guarantee 3. *For the 1-1 constraint and the cycle constraint, the runtime complexity of Algorithm 1 is* $\mathcal{O}(k \times |C|^2)$.

Proof. We start with the function *adjust*, i.e., Algorithm 2. First, all sets of correspondences that contain $\hat{c}$, but do not show θ-satisfaction are extracted. For the considered 1-1 constraint and cycle constraint, we note that the constraints in these sets are necessarily connected. Hence, the sets can be derived by depth-first-search, starting with correspondence $\hat{c}$. Visiting each correspondence at most once, this yields a runtime complexity of $\mathcal{O}(|I|)$. Moreover, there are at most $|I|$ iterations (in the worst case, all correspondences are removed). As a result, the overall complexity is $\mathcal{O}(|I|^2)$.
Finally, the most expensive operation in Algorithm 1 is the function *adjust*, which has a runtime complexity of $\mathcal{O}(|I|^2)$. Since $I \subseteq C$ and there are at most k iterations of the local search, we arrive at $\mathcal{O}(k \times |C|^2)$. □

For the structure constraint, the size of the structure relation influences the runtime complexity. However, in practice, the number of entries in the structure relation is significantly less than the number of correspondences $|R_S| \ll |C|$, so that complexity stay manageable.

Guarantee 4. *For the structure constraint, the runtime complexity of Algorithm 1 is* $\mathcal{O}(k \times |C| \times (|C| + |R_S|))$.

Proof. Similar to the proof of Guarantee 3, we also need to perform depth-first-search. The only difference is that now we have to include the relations in R_S as connections, which yields a runtime complexity of $\mathcal{O}(|I| + |R_S|)$ for the correspondence $\hat{c}$. Since remaining operations are similar, we arrive at the overall complexity $\mathcal{O}(k \times |C| \times (|C| + |R_S|))$. □

6. Experimental Evaluation

This section presents an experimental evaluation of the proposed methods to handle probabilistic constraints in matching networks, using real-world datasets and state-of-the-art matching tools. The results highlight that the presented approach supports pay-as-you-go reconciliation by effective and efficient computation of probabilities. We are able to precisely guide expert users, so that the amount of expert input needed for reconciliation is reduced to 50% or less compared to baseline solutions. We demonstrate that the approach improves the quality of instantiated matchings significantly in both precision and recall.

We proceed as follows: We first discuss the experimental setup (Section 6.1). Then, we report on the results of applying the proposed methods for probability computation (Section 6.2) and instantiation of a matching (Section 6.3).

6.1. Experimental Setup

Datasets and Matching Tools. We relied on five real-world datasets spanning various application domains, from Web forms to business schemas as observed in data marketplaces.

(1) *Business Partner (BP):* The dataset comprises database schemas that model business partners in enterprise systems.
(2) *PurchaseOrder (PO):* We extracted purchase order e-business schemas from various resources.
(3) *University Application Form (UAF):* We extracted schemas from Web interfaces of university application forms.
(4) *WebForm:* The schemas for this dataset have been automatically extracted from Web forms using OntoBuilder [45].
(5) *Conference (OAEI):* This is a collection of 16 ontologies about conference organization, from the 'conference' track of Ontology Alignment Evaluation Initiative 2014 [6].

These datasets are publicly available [4] and descriptive statistics for the models are given in Table 1. To generate candidate correspondences for schema matching datasets (BP, PO, UAF, WebForm), we used two well-known schema matchers (with default parameters), COMA++ [28] and AMC [46]. For the ontology matching datasets (OAEI), we use the AML matcher [47], due to its good performance in the OAEI [6].

Integrity Constraints. In our experiments, we consider the 1-1 constraint and the cycle constraint, as defined in Section 3. Table 2 lists the number of candidate correspondences (or sets thereof) generated by the matcher for which the constraint satisfaction probability is less than one. Rather independently of the applied dataset and matching tool, we observe a large number of problematic correspondences, which precludes an exhaustive investigation by an expert. Hence, there is a clear need for efficient and effective pay-as-you-go reconciliation.

[4] `http://lsirwww.epfl.ch/schema_matching`

Table 1: Datasets

Dataset	#Models	#Elements (Min/Max)
BP	3	80/106
PO	10	35/408
UAF	15	65/228
WebForm	89	10/120
OAEI	16	23/140

Table 2: Problematic correspondences

Dataset	# Correspondences ($v_\gamma < 1$) COMA	AMC	AML
BP	252	244	N/A
PO	10078	11320	N/A
UAF	40436	41256	N/A
WebForm	6032	6367	N/A
OAEI	N/A	N/A	9352

Evaluation Measures. We rely on the following measures:

- *Network Uncertainty* represents the overall uncertainty related to the correctness of correspondences. For a probabilistic matching network $N = \langle S, G_S, \Gamma, C, P \rangle$, it is defined as the Shannon entropy over these probabilities:

$$H(C,P) = -\sum_{c \in C} (P(c)\log P(c) + (1-P(c))\log(1-P(c))).$$

  As such, $H(C,P) = 0$ means that all probabilities assigned to candidate correspondences are equal to one or zero.
- *Precision & Recall* are measures for the quality of a matching V (i.e. a set of correspondences) compared to the exact matching R (e.g. a set of referenced correspondences given by the dataset provider): $Prec(V)=(|V \cap R|)/|V|$ and $Rec(V)=(|V \cap R|)/|R|$.
- *Expert effort:* To quantify the relative amount of expert input, we compute the effort as the number of asserted correspondences relative to the size of the matcher's output: $E = |U^+ \cup U^-|/|C|$.

Expert Guidance. To assess the effectiveness of the computed correctness probabilities, we considered two strategies to guide an expert in the validation efforts. As a baseline, we employ a random selection of correspondences for validation (*Rand*). We contrast the results obtained with this baseline with a selection of correspondences that is grounded in the information gain (*Heuristic*). That is, we select the correspondence for which expert input leads to a maximal reduction in network uncertainty, as defined above.

Experimental Environment. All results have been obtained on an Intel Core i7 system (2.8GHz, 4GB RAM). Factor graph computations have been conducted using Elementary [37].

6.2. Evaluation of the Probability Computation

To evaluate the computation of correctness probabilities of correspondences in a matching network, we first study the efficiency and effectiveness of the presented approach based on a factor graph. Finally, we also turn to the effectiveness of these probabilities to guide an expert in the reconciliation process.

Computation Time. In this experiment, we study the effects of network size (i.e., number of candidate correspondences) on the computation time required for probability computation. We use the Elementary framework [37] that computes probabilities in a factor graph by Gibbs sampling. The reported time is measured by computing the average sampling time over 1000 samples for each network size. Each setting is constructed with a different interaction graph G_S using the Erdős-Rényi random graph model. We report the average time over all settings and datasets.

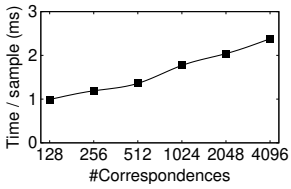

Figure 5: Effects of network size on probability computation

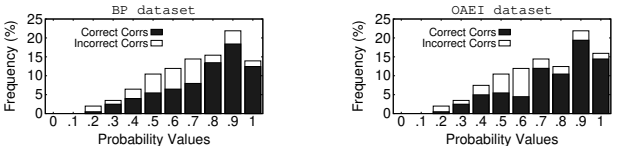

Figure 6: Relation between probability and correctness of correspondences

Figure 5 shows the resulting computation time per sample relative to the number of correspondences with values ranging from 2^7 to 2^{12}. Clearly, as the number of correspondences grows, the computation time increases. Yet, absolute numbers are low. For instance, for a network with 4000 candidate correspondences, computation based on 1000 samples takes only $\approx 2.4\text{ms} \cdot 10^3 = 2.4\text{s}$. Hence, the presented approach is well applicable for datasets with a large number of correspondences.

Relation between Probability and Correctness. Our approach is based on the hypothesis that a correspondence with high probability is likely to be correct, and vice-versa. To validate this hypothesis, we first compare the candidate correspondences with the exact matching to categorize them as correct or incorrect. Then, we compute the probability of each correspondence. Figure 6 presents a histogram of the probability distribution in the BP dataset (representative for schema matching, other datasets have similar results) and the OAEI dataset for correct and incorrect correspondences. Here, the X-axis depicts the probability ranges and the Y-axis measures the frequency in percentages.

We observe that the probability distribution of correspondences is matched well with their correctness. For example, in the BP dataset, most of the correspondences (more than 75%) have the probability value in the range from 0.5 to 1.0. This is reasonable, since the precision of the generated candidate correspondences in this dataset is about 0.67. Another key finding is that at higher levels of probability, the ratio of correct correspondences over incorrect correspondences is significantly larger. For the OAEI dataset, for instance, in the $[0.8, 0.9]$ range, there are about 19% correct and about 2% incorrect correspondences; whereas the ratio is about 14%/1% in the $[0.9, 1.0]$ range. This indicates that the probability values indeed reflect the correctness of correspondences.

Effectiveness for Expert Guidance. To explore how well the computed probabilities guide an expert in the reconciliation process, we proceed as follows. For each dataset, we generate a complete interaction graph and obtain candidate correspondences using automatic matchers. Then, we simulate the pay-as-you-go reconciliation process where expert input is generated using the exact matches, which had been constructed by the

dataset provider.

We then employ two strategies for expert guidance, random selection of correspondences (*Rand*) and selection based on the correctness probabilities in terms of a ranking of the induced information gain (*Heuristic*). The BP and OAEI datasets are used for schema matching and ontology alignment, respectively. Figure 7 and 8 depict the improvements in precision and network uncertainty (Y-axes) with increased expert effort (X-axis, given as a percentage), as an average over 50 experiment runs. The other datasets demonstrate similar results and are omitted for the sake of brevity.

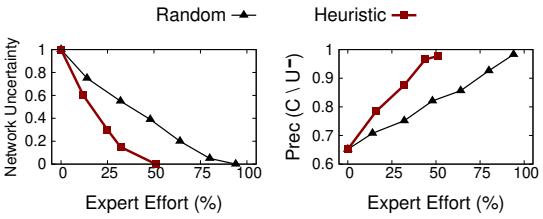

Figure 7: Expert effort needed during the reconciliation (BP dataset)

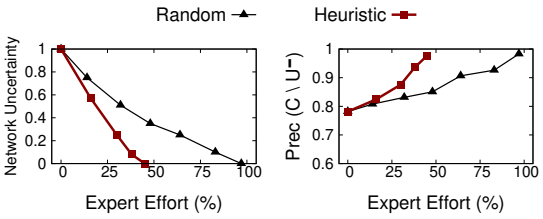

Figure 8: Expert effort needed during the reconciliation (OAEI dataset)

We observe a significant reduction of expert effort for the strategy that employs the computed correctness probabilities. More precisely, applying our solution when selecting correspondences requires only about 50% or less of the expert interactions. In terms of absolute numbers, we save 124 and 4676 validations on problematic correspondences in the BP and OAEI datasets, respectively. Another key finding is that the trends of network uncertainty and precision are inversely similar. This implies that network uncertainty is a good indicator for reconciling the matching results. Note that when network uncertainty is zero (i.e. all integrity constraints are satisfied with a probability of one), the precision is not necessarily guaranteed to be 1.0.

6.3. Evaluation of the Matching Instantiation

Next, we study the effectiveness of our method for instantiating a matching from a probabilistic matching network.

Effects of Expert Guidance. Clearly, the two above strategies to guide an expert user (i.e., *Rand* and *Heuristic*) influence the quality of the instantiated matching. We investigate this aspect with an experiment in which, given a predefined amount of expert input (e.g., validation of 5% of all candidate correspondences), we reduce network uncertainty with these strategies. Then, we compare the results in terms of precision and recall of the matching derived by instantiation (Algorithm 1).

Figure 9 and 10 show the results for the BP dataset and the OAEI dataset, respectively (again, the other datasets showed the same trend). We varied the amount of expert effort (x-axis), considering validation of 0% to 15% of the candidate correspondences. In absolute numbers, this means 0 to 38 correspondences for the BP dataset, and 1403 correspondences for the OAEI dataset. A key finding is that the guidance strategy based on the computed probabilities outperforms the baseline in terms of both precision and recall. Note that initially, with 0% expert effort, there is no difference between the two strategies, since none of the correspondences has been validated. We conclude that the computed correctness probabilities play an important role to guide the expert in the reconciliation, thereby improving the quality of the instantiated matching.

Effects of Maximal Likelihood. Instantiation is guided by the size and the likelihood of a particular matching, see Section 5. We argued that the size shall be maximal to keep as much information on correspondences as possible in the instantiated matching. Yet, in this experiment, we study the importance of also considering the likelihood of correspondences. To this end, we compare the result of instantiation with and without the likelihood criterion. We quantify the results in terms of precision and recall for the instantiated matching.

Figure 11 and 12 illustrate the percentage of expert effort relative to the precision and recall of the instantiated matching for the BP dataset and OAEI dataset, respectively. Considering the likelihood criterion indeed leads to a matching of better quality. Again, we also consider the absolute numbers: The instantiated matching has 195 correspondences and 3081 correspondences in the BP dataset and the OAEI dataset, respectively. This underlines the benefits of our probabilistic model in quantifying the uncertainty on the correctness of correspondences.

The instantiation of a matching can be interpreted in the context of data repair. Without expert guidance, all problematic correspondences that violate integrity constraints have to be assessed, including correct and incorrect ones. How- ever, using our expert guidance, only half of them are considered. Moreover, it is not only about addressing constraint violations, but also about approving correct correspondences. Put differently, our approach also gains correct information instead of only removing in- correct information during validation. For example, the matching instantiated for the OAEI dataset has 3081 correspondences, which means that around 70% of the candidate correspondences generated by automatic matchers represent “dirty” data.

7. Related Work

Below, we review work on pair-wise matching of data models, matching networks, and user feedback.

Pair-wise Matching of Data Models. Matching of database schemas is an active research field. The developments of this area have been summarized in several surveys, e.g., [3, 1, 48]. Existing work focused mainly on improving the quality parameters of matchers, such as precision or recall of the generated correspondences [49, 50, 36], or leveraging matching results for further management operations [51]. Recently, however, it was

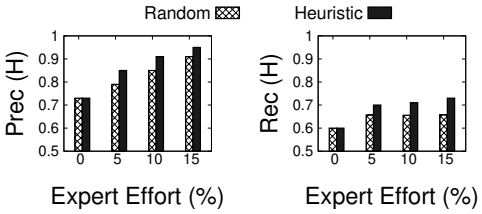

Figure 9: Effects of correspondence ordering strategies on instantiation. H is the matching instantiated by our algorithm (BP dataset)

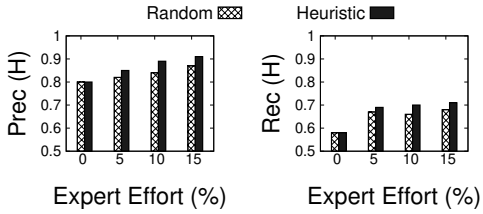

Figure 10: Effects of correspondence ordering strategies on instantiation. H is the matching instantiated by our algorithm (OAEI dataset)

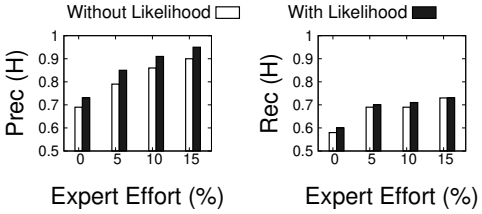

Figure 11: Effects of the likelihood function on instantiation. H is the matching instantiated by our algorithm (BP dataset)

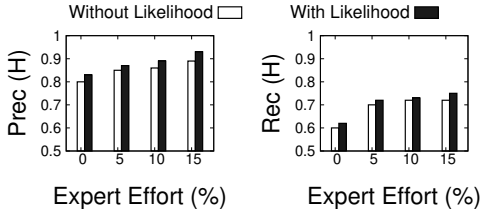

Figure 12: Effects of the likelihood function on instantiation. H is the matching instantiated by our algorithm (OAEI dataset)

realized that the extent to which precision and recall can be improved may be limited for general-purpose matching algorithms. Instead of designing new algorithms, there has been a shift towards matching combination and tuning methods. These works include YAM [52], systematic matching ensemble selection [53] or automatic tuning of the matcher parameters [54, 49].

Similar research trends and results are also found for other types of data models, such as ontologies [5, 55]. Ontology matching focuses on leveraging semantic information such as taxonomies, domain vocabularies, and resource descriptions, to improve matching techniques proposed for database schemas, which lack such semantic information [56, 57]. Despite their application for different types of data models, all these matching techniques have in common that they only consider pair-wise matching. As a consequence, they neglect a significant amount of information that materialises through dependencies of correspondences between different pairs of models.

In this paper, we use results from matching tools as input for our approach. Uncertain matching for data models has been studied extensively, for instance, [58, 59, 32, 60]. Yet, our approach is the first to consider probabilistic integrity constraints defined for a matching network to assess the correctness probabilities of correspondences and guide the reconciliation by an expert.

While our approach is based on a probabilistic model, possibility theory [61, 62, 63, 64] provides an orthogonal approach for matching computation. While the choice of the underlying model depends on the application domain, our motivation for a probabilistic model is threefold. First, as mentioned above, state-of-the-art matchers for database schemas and ontologies produce probabilistic values as a confidence level for their candidate correspondences. Second, probability theory enables us to handle soft constraints with a single configurable parameter. Third, possibility theory assumes a definition of states or value ranges of the likelihood. Such a definition is challenging in the domains of schema matching and ontology alignment. At the same time, we expect it to be difficult for users to interpret results that declare correspondences, e.g., to be 'unlikely to be true'. Despite these challenges, a possibility theory may provide a complementing angle on the reconciliation of correspondences that shall be explored in future research.

Matching Networks. The idea of exploiting a set of data models as a whole to improve the matching has been studied before. Holistic matching [65] attempted to exploit statistical co-occurrences of attributes in different database schemas and use them to derive complex correspondences. Corpus-based matching [66] uses a 'corpus' of schemas to augment the evidence that improves existing matching and exploit constraints between attributes by applying statistical techniques. Nevertheless, these approaches follow a mediated approach, which constructs one data model as a single-point reference for all original models. The approach is also studied in more fine-grained settings such as LAV and GAV [1]. While LAV focuses on matching from source models to the mediated model, GAV generates matching from the mediated model to the source models. Mediated approaches, however, have two limitations: They assume that (1) it is possible to develop a consensus model that captures all different characteristics of original models, (2) it is feasible to update the mediated model when the original ones are altered.

Several studies have shown that there are application scenarios, for which it is infeasible to develop and maintain such a consensus model, see [20, 67, 68]. A core problem here is that matching networks tend to be dynamic: Organisations and, therefore, data models, join and leave the network frequently. An example are the matching networks developed by the NisB project (funded by the European Commission and involving SAP and FIAT as industrial partners) that to enable the exchange of documents within a flexible, market-oriented business network. In fact, the *Business Partner* and *PurchaseOrder* datasets used in our evaluation stem from this project.

Going beyond the state-of-the-art, our work targets such settings, where the computation with a monolithic, mediated

model is too costly or simply infeasible. It also acknowledges that a plethora of algorithms for pairwise model matching have been developed, so that pairwise matchings are a viable starting point for data integration. The network setting then enables us to incorporate transitive properties (i.e. integrity constraints) that help to select trusted matches and minimise user effort in validating correspondences.

Network-level integrity constraints. Network-level constraints were originally considered in [69, 12], in which the establishment of semantic interoperability in large-scale P2P networks was studied. As discussed earlier, these ideas have been adopted for integrity constraints in the reconciliation process of database schemas in [11, 13, 70]. Yet, these approaches put forward a simplistic view on data models and, thus, integrity constraints.

In prior work, we presented the first approach to incorporate network-level integrity constraints for data models [11]. However, in this work, we target a more general setting and adopt more expressive notions of data models and constraints, i.e., models have an internal structure and integrity constraints may be probabilistic. These more expressive notions motivated our technical contributions on the probability computation and the matching instantiation, since the techniques developed in [11] are no longer applicable. Note that the differences in expressiveness of the constraints precludes any direct comparison of the techniques for discrete constraints and probabilistic constraints.

User Feedback. The post-matching reconciliation process has received considerable attention in the literature for schemas and ontologies. The systems in [7, 8, 9, 71, 72, 73] rely on one user only, whereas the frameworks in [74, 75, 36, 76, 10, 35, 77, 78] rely on multiple users. Although we focus on reconciliation with a single expert, our framework is extensible as the underlying probabilistic model is independent of the number of users.

The idea of pay-as-you-go approaches to improve the matching quality has been brought forward in [59]. However, we note that the approach in [59] requires the creation of a mediated model, whereas we study reconciliation for a network of data models. Moreover, unlike many existing works, e.g., [8, 9] that incorporate user feedback implicitly through keywords, our approach lets experts give input explicitly on the correspondences. This results in a clear quantification of expert effort.

Reconciliation of correspondences based on network-level constraints has been proposed for the first time in [11]. However, in this paper, we ground reconciliation in more expressive formalisms for data models and constraints, incorporating the structure of data models and supporting the definition of soft constraints to cope with models of varying abstraction levels. These more expressive formalisms render the techniques for probability computation and matching instantiation presented in [11] inapplicable. We therefore proposed a novel approach for probability computation based on a factor graph and introduced a new instantiation problem that is grounded in the notion of θ-satisfaction of integrity constraints.

8. Conclusions and Future Work

In this paper, we presented models and methods to support pay-as-you-go reconciliation of matching networks based on consistency expectations that are formulated as integrity constraints. Our focus have been data model show some internal structure and, potentially, incorporate different abstractions of some data entities. We argued that, for such a setting, a probabilistic model of integrity constraints is needed. We proposed such a model and developed a representation as a factor graph in order to compute correctness probabilities of correspondences. Most importantly, our probabilistic graphical model enables us to capture the uncertainty in the matching network in a unified way, integrating the output of automatic matching, expert input, and the consistency expectations in terms of integrity constraints. We further introduced a method instantiate a trusted set of correspondences. As such, the approach can be used to support the handling of data models at any point in time, while still continuously improving the quality of the instantiated matching by reconciliation of the network. Our empirical evaluation further demonstrated the effectiveness and efficiency of the presented approach. It is applicable to large, real-world datasets, while the computed correctness probabilities guide an expert user effectively in the reconciliation.

Our contributions open up several future research directions. The proposed probabilistic formulation of constraints can be extended to further develop the quality measurement of matching networks. Moreover, on top of matching networks, we intend to develop a wide range of potential utilities in data management systems. Examples of such utilities include visualizing a matching network at large scales, searching & filtering network-level information and data sources efficiently, and reconciling the network dynamically when a new data source arrives.

[1] P. Bernstein, J. Madhavan, E. Rahm, Generic Schema Matching, Ten Years Later, in: VLDB, 2011.

[2] P. Shvaiko, J. Euzenat, Tutorial on schema and ontology matching, in: ESWC, 2005, pp. 05–29.

[3] E. Rahm, P. A. Bernstein, A Survey of Approaches to Automatic Schema Matching, JVLDB (2001) 334–350.

[4] M. G. De Carvalho, A. H. Laender, M. A. GonçAlves, A. S. Da Silva, An evolutionary approach to complex schema matching, Information Systems 38 (3) (2013) 302–316.

[5] N. F. Noy, M. A. Musen, Algorithm and tool for automated ontology merging and alignment, in: AAAI, 2000.

[6] Z. Dragisic, K. Eckert, J. Euzenat, A. Ferrara, R. Granada, V. Ivanova, E. Jiménez-Ruiz, A. O. Kempf, P. Lambrix, et al., Results of the ontology alignment evaluation initiative 2014.

[7] S. R. Jeffery, M. J. Franklin, A. Y. Halevy, Pay-as-you-go user feedback for dataspace systems, in: SIGMOD, 2008, pp. 847–860.

[8] M. Sayyadian, H. LeKhac, A. Doan, L. Gravano, Efficient keyword search across heterogeneous relational databases, in: ICDE, 2007, pp. 346–355.

[9] Z. Yan, N. Zheng, Z. G. Ives, P. P. Talukdar, C. Yu, Actively soliciting feedback for query answers in keyword search-based data integration, in: PVLDB, 2013, pp. 205–216.

[10] K. Belhajjame, N. W. Paton, A. A. A. Fernandes, C. Hedeler, S. M. Embury, User feedback as a first class citizen in information integration systems, in: CIDR, 2011, pp. 175–183.

[11] Q. V. H. Nguyen, T. T. Nguyen, Z. Miklós, K. Aberer, A. Gal, M. Weidlich, Pay-as-you-go reconciliation in schema matching networks, in: ICDE, 2014, pp. 220–231.

[12] P. Cudré-Mauroux, K. Aberer, A. Feher, Probabilistic Message Passing in Peer Data Management Systems, in: ICDE, 2006, pp. 41–52.

[13] Q. V. H. Nguyen, T. K. Wijaya, Z. Miklos, K. Aberer, E. Levy, V. Shafran, A. Gal, M. Weidlich, Minimizing Human Effort in Reconciling Match Networks, in: ER, 2013.

[14] T. Sagi, A. Gal, Non-binary evaluation measures for big data integration, VLDBJ (2018) 105–126.

[15] B. Gu, Z. Li, X. Zhang, A. Liu, G. Liu, K. Zheng, L. Zhao, X. Zhou, The interaction between schema matching and record matching in data integration, TKDE (2017) 186–199.

[16] Y. Wang, Y. He, Synthesizing mapping relationships using table corpus, in: SIGMOD, 2017, pp. 1117–1132.

[17] P. Vassiliadis, A. V. Zarras, I. Skoulis, Gravitating to rigidity: Patterns of schema evolution–and its absence–in the lives of tables, Information Systems 63 (2017) 24–46.

[18] H. Gonzalez, A. Y. Halevy, C. S. Jensen, A. Langen, J. Madhavan, R. Shapley, W. Shen, J. Goldberg-Kidon, Google fusion tables: web-centered data management and collaboration, in: SIGMOD, 2010, pp. 1061–1066.

[19] W. Litwin, L. Mark, N. Roussopoulos, Interoperability of multiple autonomous databases, CSUR (1990) 267–293.

[20] K. Smith, M. Morse, P. Mork, M. Li, A. Rosenthal, D. Allen, L. Seligman, C. Wolf, The role of schema matching in large enterprises, in: CIDR, 2009.

[21] K. Aberer, P. Cudre-Mauroux, A. M. Ouksel, T. Catarci, M.-S. Hacid, A. Illarramendi, V. Kashyap, M. Mecella, E. Mena, E. J. Neuhold, O. D. Troyer, T. Risse, M. Scannapieco, F. Saltor, L. D. Santis, S. Spaccapietra, S. Staab, R. Studer, Emergent semantics principles and issues, in: DASFAA, 2004, pp. 25–38.

[22] L. Ardissono, A. Goy, G. Petrone, M. Segnan, From service clouds to user-centric personal clouds, in: CLOUD, 2009, pp. 1–8.

[23] S. R. Yerva, Z. Miklós, K. Aberer, Quality-aware similarity assessment for entity matching in web data, Information Systems 37 (4) (2012) 336–351.

[24] D. Rinser, D. Lange, F. Naumann, Cross-lingual entity matching and infobox alignment in wikipedia, Information Systems 38 (6) (2013) 887–907.

[25] D. Ritter, N. May, S. Rinderle-Ma, Patterns for emerging application integration scenarios: A survey, Information Systems 67 (2017) 36–57.

[26] D. Fensel, H. Lausen, A. Polleres, J. de Bruijn, M. Stollberg, D. Roman, J. Domingue, Enabling semantic web services: the web service modeling ontology, Springer, 2006.

[27] M. Paolucci, T. Kawamura, T. R. Payne, K. Sycara, Semantic matching of web services capabilities, in: ISWC, 2002, pp. 333–347.

[28] D. Aumueller, H.-H. Do, S. Massmann, E. Rahm, Schema and ontology matching with coma++, in: SIGMOD, 2005, pp. 906–908.

[29] D. H. Ngo, Z. Bellahsene, YAM++ : (not) Yet Another Matcher for Ontology Matching Task, in: BDA, 2012.

[30] L. Seligman, P. Mork, A. Halevy, K. Smith, M. J. Carey, K. Chen, C. Wolf, J. Madhavan, A. Kannan, D. Burdick, Openii: an open source information integration toolkit, in: SIGMOD, 2010, pp. 1057–1060.

[31] A. Doan, P. Domingos, A. Y. Halevy, Reconciling schemas of disparate data sources: a machine-learning approach, in: SIGMOD, 2001, pp. 509–520.

[32] A. Gal, Uncertain Schema Matching, Morgan & Claypool, 2011.

[33] A. Gal, T. Sagi, M. Weidlich, E. Levy, V. Shafran, Z. Miklós, N. Hung, Making sense of top-k matchings: A unified match graph for schema matching, in: IIWeb, 2012.

[34] F. R. Kschischang, B. J. Frey, H.-A. Loeliger, Factor graphs and the sum-product algorithm, IEEE Trans. Inf. Theory (2001) 498–519.

[35] Q. V. Nguyen, H. X. Luong, Z. Miklós, K. Aberer, Collaborative schema matching reconciliation, in: CoopIS, 2013.

[36] Q. V. H. Nguyen, T. T. Nguyen, Z. Miklós, K. Aberer, On leveraging crowdsourcing techniques for schema matching networks, in: DASFAA, 2013, pp. 139–154.

[37] C. Zhang, C. Ré, Towards high-throughput gibbs sampling: A study across storage managers, in: SIGMOD, 2013, pp. 397–408.

[38] A. McCallum, K. Bellare, F. Pereira, A conditional random field for discriminatively-trained finite-state string edit distance, in: UAI, 2005, pp. 388–395.

[39] G. Casella, E. I. George, Explaining the gibbs sampler, The American Statistician 46 (3) (1992) 167–174.

[40] C.-J. Kim, C. R. Nelson, et al., State-space models with regime switching: classical and gibbs-sampling approaches with applications, MIT Press Books 1.

[41] B. Marthi, H. Pasula, S. Russell, Y. Peres, Decayed mcmc iltering, in: UAI, 2002, pp. 319–326.

[42] J. Hopcroft, R. Tarjan, Algorithm 447: efficient algorithms for graph manipulation, CACM (1973) 372–378.

[43] Q. V. H. Nguyen, Reconciling schema matching networks, Ph.D. thesis, EPFL (2014).

[44] M. Franklin, A. Halevy, D. Maier, From databases to dataspaces: a new abstraction for information management, in: SIGMOD, 2005, pp. 27–33.

[45] H. Roitman, A. Gal, Ontobuilder: fully automatic extraction and consolidation of ontologies from web sources using sequence semantics, in: EDBT, 2006, pp. 573–576.

[46] E. Peukert, J. Eberius, E. Rahm, AMC - A framework for modelling and comparing matching systems as matching processes, in: ICDE, 2011, pp. 1304–1307.

[47] D. Faria, C. Pesquita, E. Santos, I. F. Cruz, F. M. Couto, Agreementmakerlight: a scalable automated ontology matching system, in: DILS, 2014, p. 29.

[48] Z. Bellahsene, A. Bonifati, E. Rahm, Schema Matching and Mapping, Springer, 2011.

[49] A. Gal, H. Roitman, T. Sagi, From diversity-based prediction to better ontology & schema matching, in: WWW, 2016, pp. 1145–1155.

[50] C. J. Zhang, L. Chen, H. Jagadish, C. C. Cao, Reducing uncertainty of schema matching via crowdsourcing, in: VLDB, 2013, pp. 757–768.

[51] T. T. Nguyen, Q. V. H. Nguyen, M. Weidlich, K. Aberer, Result selection and summarization for web table search, in: ICDE, 2015, pp. 231–242.

[52] F. Duchateau, R. Coletta, Z. Bellahsene, R. J. Miller, (not) yet another matcher, in: CIKM, 2009, pp. 1537–1540.

[53] A. Gal, T. Sagi, Tuning the ensemble selection process of schema matchers, JIS (2010) 845–859.

[54] Y. Lee, M. Sayyadian, A. Doan, A. S. Rosenthal, eTuner: tuning schema matching software using synthetic scenarios, JVLDB (2007) 97–122.

[55] M. Vargas-Vera, M. Nagy, State of the art on ontology alignment, IJKSR (2015) 17–42.

[56] L. Otero-Cerdeira, F. J. Rodríguez-Martínez, A. Gómez-Rodríguez, Ontology matching: A literature review, ESWA (2015) 949–971.

[57] N. T. Tam, M. Weidlich, D. C. Thang, H. Yin, N. Q. V. Hung, Retaining data from streams of social platforms with minimal regret, in: IJCAI, 2017, pp. 2850–2856.

[58] X. Dong, A. Y. Halevy, C. Yu, Data integration with uncertainty, in: PVLDB, 2007, pp. 687–698.

[59] A. D. Sarma, X. Dong, A. Y. Halevy, Bootstrapping pay-as-you-go data integration systems, in: SIGMOD, 2008, pp. 861–874.

[60] J. Gong, R. Cheng, D. W. Cheung, Efficient management of uncertainty in XML schema matching, JVLDB (2012) 385–409.

[61] N. B. Amor, D. Dubois, H. Gouider, H. Prade, Possibilistic preference networks, ISCI 460 (2018) 401–415.

[62] M. Haddad, P. Leray, N. B. Amor, Learning possibilistic networks from data: a survey., in: IFSA-EUSFLAT, 2015.

[63] C. Borgelt, R. Kruse, Operations and evaluation measures for learning possibilistic graphical models, AI 148 (1-2) (2003) 385–418.

[64] C. Borgelt, R. Kruse, Learning possibilistic graphical models from data, TFS 11 (2) (2003) 159–172.

[65] W. Su, J. Wang, F. Lochovsky, Holistic schema matching for web query interfaces, in: EDBT, 2006, pp. 77–94.

[66] J. Madhavan, P. A. Bernstein, A. Doan, A. Halevy, Corpus-based schema matching, in: ICDE, 2005, pp. 57–68.

[67] B. Saha, I. Stanoi, K. L. Clarkson, Schema covering: a step towards enabling reuse in information integration., in: ICDE, 2010, pp. 285–296.

[68] A. Gal, M. Katz, T. Sagi, M. Weidlich, K. Aberer, N. Q. V. Hung, Z. Miklós, E. Levy, V. Shafran, Completeness and ambiguity of schema cover, in: CoopIS, 2013, pp. 241–258.

[69] K. Aberer, P. Cudré-Mauroux, M. Hauswirth, Start making sense: The Chatty Web approach for global semantic agreements, JWS (2003) 89–114.

[70] T. T. Nguyen, T. C. Phan, Q. V. H. Nguyen, K. Aberer, B. Stantic, Maximal fusion of facts on the web with credibility guarantee, Information Fusion 48 (2019) 55–66.

[71] S. Duan, A. Fokoue, K. Srinivas, One size does not fit all: Customizing ontology alignment using user feedback, in: ISWC, 2010, pp. 177–192.

[72] I. F. Cruz, C. Stroe, M. Palmonari, Interactive user feedback in ontology matching using signature vectors, in: ICDE, 2012, pp. 1321–1324.

[73] N. Q. V. Hung, D. C. Thang, M. Weidlich, K. Aberer, Minimizing efforts in validating crowd answers, in: SIGMOD, 2015, pp. 999–1014.

[74] A. V. Zhdanova, P. Shvaiko, Community-Driven Ontology Matching, in: ESWC, 2006, pp. 34–49.
[75] R. McCann, W. Shen, A. Doan, Matching Schemas in Online Communities: A Web 2.0 Approach, in: ICDE, 2008, pp. 110–119.
[76] I. F. Cruz, F. Loprete, M. Palmonari, C. Stroe, A. Taheri, Pay-as-you-go multi-user feedback model for ontology matching, in: EKAW, 2014, pp. 80–96.
[77] N. Q. V. Hung, N. T. Tam, L. N. Tran, K. Aberer, An evaluation of aggregation techniques in crowdsourcing, in: WISE, 2013, pp. 1–15.
[78] N. Q. V. Hung, C. T. Duong, N. T. Tam, M. Weidlich, K. Aberer, H. Yin, X. Zhou, Argument discovery via crowdsourcing., VLDB J. 26 (4) (2017) 511–535.